IoT 班成果報告書

井上諒也 *
山口流星 †
青木雅典 ‡
水野佑哉 §
安田弘樹 ¶

平成 30 年 2 月 5 日

*理工学部 都市システム工学科 二回生
†理工学部 数理科学科 一回生
‡理工学部 電子情報工学科 一回生
§理工学部 ロボティクス学科 二回生
¶情報理工学部 知能情報学科 二回生

目 次

1　はじめに

文責：井上　諒也

人間はかねてより生活環境の向上にという題に対して最先端の技術を駆使してきた. 現代において急速に発達している情報通信技術は我々が生きる人間社会において大きな要となる. 情報通信技術を生かしたものの中に「Internet of Things」という概念がある. 「Internet of Things」(以下, IoT) とは家電製品や産業機械等といった現実社会を構成している「モノ」が, パソコンやサーバーが属しているインターネットに接続し,「モノ」から収集したデータに応じてアクションを実行するという概念である. 本プロジェクトでは, IoT に関する知識の向上と人間社会の効率化を図るためのアイデアの発案, そのアイデアを実現する為のシステム設計, 実装を行うための技術力の向上を目的とした.

2　IoT について

2.1　IoT の現状

文責：井上　諒也

現在, IoT が経済的に大きな効果をもたらしていることから, 世界中から将来を担う技術の一つとして注目されている. 世界各国の研究者が Industry 4.0(第四の産業革命) や次世代技術戦略プロジェクトとして推進された「IoT」について共同研究を行ってきたが, 現代では「IoT」という言葉や概念は一般層にまで普及している. 「IoT」を構築するためのサービスやデバイスの操作は分かりやすい仕様のものが増加しており, システムの実装は以前までと比べて大きく敷居が低くなった. そのために現在では身近な生活環境をよりよくするために活用され, 中でもそのアイデアは最も重要な要素の一つとされている.

2.2　IoT の背景

文責：安田　弘樹

IoT を実装するための Web サービスやプロダクト等は多数存在する. そこで, この節では主に, IoT の実装を補助する二つの Web サービスの紹介や, それらの活用について述べる.

myThings

スマートフォン用のアプリケーション[*1]. Gmail や Twitter, Facebook 等の既存の Web サービス, IoT プロダクト等を二つ選択して組み合わせる事により, IoT と接続しているデバイス, つまりスマートフォンの所有者の行動や天気の状態, 時間と言ったようなモノの情報を取得して指定した Web サービスを作動させる事が出来る. 二つのサービスとプロダクトを繋げた機能が簡便に作成出来る事が利点であり, デバイスに独自の動作を行わせる事が可能である.

IFTTT

Web サービス[*2]. Evernote や Instagram, Dropbox のような Web サービスを連携させ, 新たなサービスを作成する事が可能. 大まかなサービス内容は前述の myThings と似通っているが, myThings と違い Windows のようなパソ

[*1] https://mythings. yahoo.co.jp/
[*2] https://ifttt.com/

コン用の OS からでも操作出来, 有料となる代わりに作成したサービスを自社製品へ組み込む事や複数の出力が可能という利点が存在する.

サービスの活用

上記二つのサービスを用いた IoT の活用例を挙げると, IoT デバイスとして使用しているスマートフォンを介して所有者の位置情報を取得. 駅や会社, 学校と行ったような特定の地域から所有者が離れた際にアクションを発生させ, 家族の携帯電話にメールやメッセージを送信し, 近いうちに帰宅する旨を自動で連絡するといった動作を行う事が可能である.

デバイスと IoT

IoT の実装を補助するツールには, 前述した二つの Web サービス以外にも MESH[*3] や MAMORIO[*4] といったように多彩な物が存在する. それらを用いる事により, 携帯電話や PC を介して一般家電の操作も可能となる. MESH の詳細は後述の項目にて記すが, MESH を用いた IoT の活用例を挙げると, IoT デバイスの所有者が帰宅を始めると連絡すれば, それに合わせて事前に部屋に配置されているロボットが部屋の湿度や温度を計測する. 計測数値が指定の数値よりも上下している場合はエアーコンディショナーや空気清浄機を起動し, デバイス所有者が帰宅するまでに部屋の状態を快適な物へと整えておくといった扱い方が可能となる. このように, IoT を活用する事により, 既存の家電製品の機能性を拡張する事も可能である.

[*3] https://meshprj.com/jp/
[*4] https://mamorio.jp/

3　MESH について

3.1　MESH とは

文責：山口　流星

MESH は「つくれる, 学べる, 楽しめるアイデアを形にできる IoT ブロック」[*5]を主軸に置いたハードウェアである. SONY から発売されたこの IoT ブロックは, MESH タグと呼び, 消しゴムほどの大きさの無線電子タグである. 現在[*6]七種類のタグがあり, 以下の画像[*7]を載せる.

- LED
- ボタン
- 人感
- 動き
- 湿度・温度
- 明るさ
- GPIO

図 1: MESH(左から上記順)

これらのタグをタブレットやスマートフォン等の電子端末と Bluetooth で通信することで使用する. この SONY の MESH タグの最大の魅力は, 使用者はこのタグの仕様に関する知識（ハードウェアの内部）及び高度なプログラミング技術を必要としないというところにある. アイデアさえあれば すぐに作成することができ, また後述の MESH SDK[*8]を使うと JavaScript を用いて高度なアイデアを実現することも可能である.

[*5] http://meshprj. com/jp
[*6] 2018 年 2 月 5 日
[*7] http://meshprj.com/jp
[*8] 3.3 を参照

3.2 MESH アプリ

文責：山口　流星

MESH タグを使うためには専用のアプリと電子端末が必要である．MESH が対応している電子端末は, Android, IOS を搭載したスマートフォン, タブレット等で, Google Store, App Store から入手することができ, MESH はこのアプリを使用して実装することになる．しかし端末によって対応していない, もしくは機能が制限される場合がある．

このアプリの役割は, MESH を動かすためのレシピを作成し, 実行することである．レシピを作ることは非常に簡単で, MESH タグとギミックをレシピ上で GUI 操作で繋いでいくのみである．ギミックは, カメラやマイク, ミュージックなどの電子端末の機能を使うものから, 論理的な回路を組むための AND やスイッチ, カウンターなどがある．さらに, 外部サービスである IFTTT[*9]や LINE, Twitter などの SNS サービスとも使えて, さらには高度であるが, ギミックを自作することも可能である．アプリはバックグラウンド処理にも対応しており, Bluetooth 通信ができればアプリを動かすことは可能である．なお, 現在はアップデートにより, MESH ハブアプリケーションとして, Raspberry Pi と連携して使用できるようになった．

3.3 MESH の通信規格

文責：青木 雅典

MESH タグは, 電子端末との通信に,「Bluetooth Low Energy（BLE)」と呼ばれる通信規格を採用している．BLE では, Peripheral/Central と呼ばれる Client/Server アーキテクチャを Attribute Protocol（ATT）というプロトコルに従って実現している．また, データ構造とやりとりの方法を定義した「Generic Attribute Profile（GATT)」と呼ばれる．仕様に基づいてアプリケーションを構築することで, 異なるメーカーの機器でもデータのやり取りが可能となる．

3.4 MESH 開発の可能性

文責：青木 雅典

MESH タグは, 現状では専用アプリで作成したレシピを元に動作するため, 対応端末以外では開発することができない．しかし, MESH タグは BLE の GATT を採用しているため, 専用アプリに対応した端末以外でも接続するこ

[*9] 2. 1 参照

とは可能である．そこで今回は, MESH タグのうち比較的動作が単純であるボタンタグについて, 専用アプリを使用しない接続での動作検証を行った．

今回は, Node.js のモジュールとして用意された「noble」を利用して検証アプリを製作し, Bluetooth 4.2 に対応した Macbook Pro 上で検証を行った．MESH タグは BLE の Peripheral として動作するため, 検証ソフトは Central として動作させた．この結果, ボタンタグの「タップ」「長押し」「ダブルタップ」の 3 種類の動作に対して, それぞれ異なる値が Central に通知されることを確認できた．

これは, MESH 専用アプリの動作しない Macbook Pro 上でも, ボタンタグの状態を取得できたことを意味している．これを活用すると, 今後は専用アプリのレシピだけでなく, 開発者がプラットフォームを自由に選択することができ, より発展的な開発が可能となると考える．

3.5 MESH Software Development Kit

文責：井上　諒也

MESH には七つのタグがあるが, これ以外にも開発者が独自でソフトウェアタグを作成し, 使用することを可能にする Software Development Kit(以下, SDK と記載する) がある．以下の図が MESH の SDK の画面であり, ソフトウェアタグの作成手順通りに作業を進めていくと, 簡単にソフトウェアタグを作成できる．

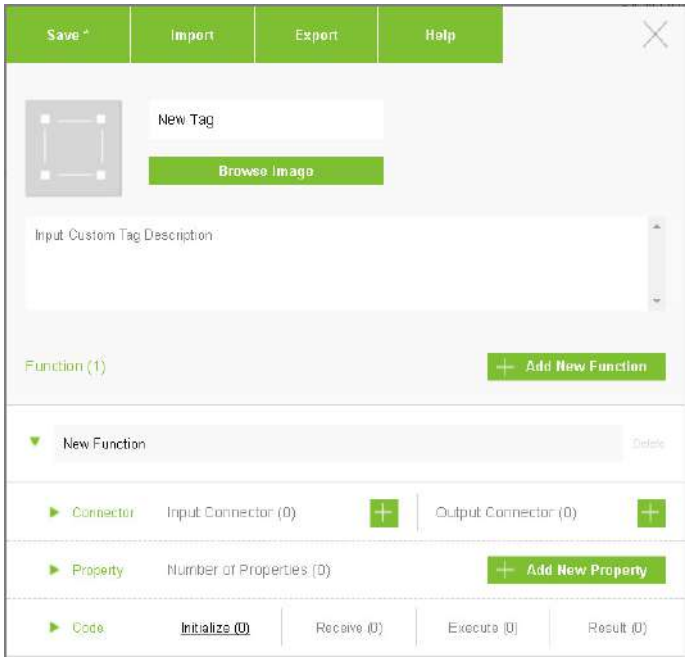

図 2: MESH SDK

タグの更新時には, MESH SDK の画面でソフトウェアタグの仕様を変更し保存する. 次に MESH アプリ上でソフトウェアタグのアップデートを行うと, ソフトウェアタグは随時更新可能となる. MESH タグには入力コネクタと出力コネクタの少なくとも一つが存在しているが, SDK ではコネクタの数を選択することが可能である. 独自タグの入力コネクタを作成することで出力コネクタを持つタグと, 出力コネクタを作成することで入力コネクタを持つタグと接続することができる.

SDK ではそれぞれ以下の 4 つのプロセスにおいて, JavaScript 言語でコードを書くことによりソフトウェアタグ内部の実装をすることができる.

- Initialize
- Receive
- Execute
- Result

「Initialize」のプロセスでは, 初期化処理の設定を行う. 「Recieve」のプロセスでは, 複数のコネクタを持つ際に, 各コネクタの入力値に対する処理を記述する. 「Excute」のプロセスでは, プログラムの大本となる処理を示す. 「Result」では出力コネクタから次のタグへ送信する際に実装する処理を示す.

一見, 難解そうに見えるかもしれないが, MESH の公式ページ[*10]にて活用例とサンプルコードが上がっている. 今回は上記の記述を元に WeatherAPI から立命館大学周辺のお天気データの取得する, など実際に制作を通しながら, ソフトウェアタグ作成の学習を行った.

3.6 Raspberry Pi とハブアプリケーション

文責：井上　諒也

MESH を用いるには通常, タブレット端末と MESH タグを Bluetooth 接続する必要がある事は前述した通りである. しかし IoT システムを構築する際に, タブレット端末を MESH タグと常時, 接続状態にする事は非常に不便である. その為, 2017 年 12 月末より Raspberry Pi と MESH タグを接続し, Raspberry Pi をハブとして動作させることが可能になった. ハブとしての動作に対応するには OS が Rspbian 対応の Raspberry Pi にハブアプリケーションを導入する必要がある. 導入後, 初期設定ではタブレット端末と Raspberry Pi の Bluetooth 接続と MESH レシピを送信するためにインターネット接続を行う必要はあるが, それ以外ではタブレット端末を用いらず MESH レシピを実行することができるようになる. このアプリケーションを用いることで, IoT システム開発について利便性が大きく増した.

[*10] https://meshprj.com/sdk/doc/ja/

4　IoT を用いたシステムの実装

4.1　目的

文責：井上　諒也

立命館コンピュータクラブの部室の構造上の問題として挙げられるのは,「窓」が無いということである. 外の環境が分からないままで室内と室外の間のギャップを感じる事がある. 悪天候になり始めた時, 部室に「窓」がある場合だと雨が降ることを予測できるが, 窓がない場合は全く気付くことができない. また景色には快適さと開放感を与えてくれる作用もある. デジタルの窓を設置することで部室の閉塞感を改善できるのではないか. この問題を IoT と関連させてシステムの実装を行うこと目的とした.

4.2　本システムの全体像

文責：井上　諒也

本システムを構築するにあたって, 重点においた要素は以下で挙げた三点である.

- 降水量や温度, 湿度などのデータの取得
- インターネットを介した通信
- データに応じて変化する窓

ハードウェア部分では MESH により温度と湿度, 照度の測定を行い, センサーモジュールを用いて降水量の推定を行う. 今回は取得したデータを HTTP 通信を用いて, 構築した Web サーバ上のデータベースに格納し, データによって「窓の外の景色」を表示する仕様である. また,「窓の外の景色」の実装は外の景色を彷彿とさせるデザインの Web ページをタブレット型端末に表示するという仕様である. 今回のシステムを構築するにあたって, 全体像を示した図を以下に載せた.

4.3　ハードウェアの実装

文責：井上　諒也

本章ではデータを取得するためのセンサー機構と MESH タグの連携についての説明を行う. データは天候の予測や気温や湿度の計測, 降水量の予測を中心にセンサーを設置した.

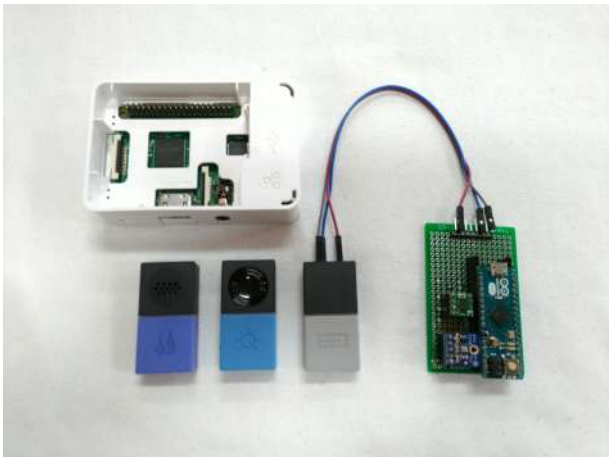

図 3: ハードウェアの全体像

4.3.1 MESH タグを用いたデータ取得

MESH タグは三つのタグを用いて温度・湿度, 照度を計測した. それぞれのタグと計測データ, 計測理由は以下の表に記載した. これらのタグを前項[*11]で述べた通り, Raspberry Pi をハブとして動作させ, Web サーバにデータを送信する. MESH レシピは以下の図の通りである.

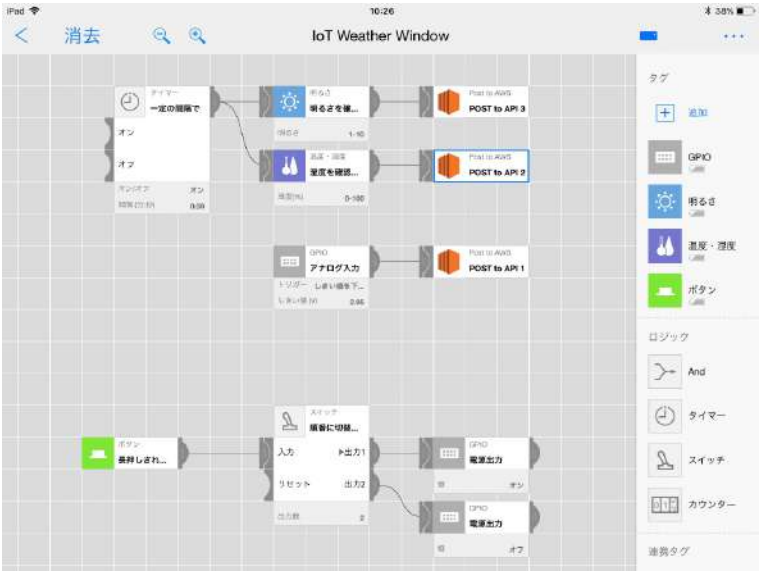

図 4: MESH のレシピ

[*11]Raspberry Pi とハブアプリケーション

ソフトウェアタグ「AWS EC2」では, 入力コネクタを各 MESH タグの出力コネクタと接続している. 各タグが測定しているデータを取得する変数を用いて, データを JSON 形式に変換し, Web サーバーへ送信している.

4.3.2　センサーモジュールによるデータ取得

センサーは小型傘の裏側に設置し, 雨の衝撃と衝撃音で降水の度合いを計測する. 降水量データを評価する為に, 加速度センサーモジュールとマイクモジュールを用いた. センサーモジュールを用いることで, MESH タグにはないデータを取得することでき幅が広がる. 使用したデバイスは以下に記した.

- モバイルバッテリー
- GPIO タグ
- MMA7361(アナログ加速度センサーモジュール)
- SPW2430 搭載のシリコン MEMS マイクモジュール
- Arduino micro

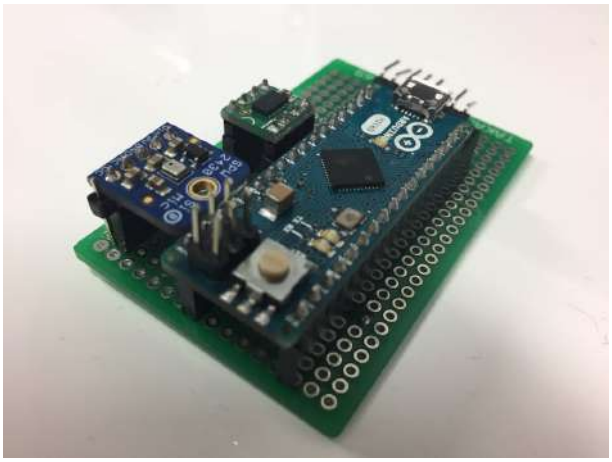

図 5: センサーと Arduino Micro

両センサーから取得した値を電圧値に変換させ, GPIO タグのアナログ入力ピンを通じて電圧値を知らせる仕様である. 閾値よりも小さい値をとると加速度センサーの値を出力させ, 大きい値の場合にマイクモジュールの値を出力させることによって, 強雨が降り注ぎ振動により判定ができなくなるとき, 雨の衝撃音での判定に切り替えることができる. GPIO タグより用意された変数で値を取得し, MESH による計測の項で述べたように JSON 形式に変

換し, Web サーバへ送信する. 今回はシステム構築を目的としたため, ハードウェア側においてはセンサー基板の作成から Web サーバーへデータを送信するまでの流れを実装した .

4.4 Web ページの制作

文責：山口　流星

4.4.1 概要

Web ページ制作にあたり必要となる Web サーバの構築, HTML ファイルの作成の二点に関して仕様の詳細を述べていく.

4.4.2 Amazon Web Services

Amazon Web Services (以下, AWS）とは, Amazon[*12]が提供しているクラウドコンピューティングサービスのことである. クラウドコンピューティングの利点として, 高度な IT 資産を運用せずに, 必要に応じて不足しているインフラをクラウドサービスで整備することができる. また, 使用した分だけの費用を支払うことが特徴なので, ストレージ量などは必要に応じて即時に対応させることが出来る, などが挙げられる. なお, AWS に新規登録をすると, 現在[*13]一年間無料で一定の範囲までサービスを利用することが出来る.

今回はこの AWS のサービスの一つ, Amazon Elastic Compute Cloud（以下, EC2）を使って, Web サーバを立てることに使用した. 実装当初では, Amazon Kinesis Data Streams, Amazon Simple Storage Service(S3), Amazon API Gateway などのサービスを使うことも予想していたが, 実装完了までに使用したサービスは EC2 のみになった.

4.4.3 EC2 と Web サーバ

Amazon Elastic Compute Cloud（EC2）とは, 仮想サーバを立てることが出来るサービスである. EC2 のサービスでは, まず仮想コンピューティングをするためのインスタンスを生成する. インスタンスは, オペレーティングシステムやソフトウェアを追加し, CPU やメモリ, ストレージやセキュリティグループを GUI 操作で選択する.[*14] インスタンスを生成したのち, Web サーバを立てるための環境構築をしていく. 言語には Ruby を採用した. 使用し

[*12] https://aws.amazon.com/jp
[*13] 2018 年 2 月 3 日
[*14] セキュリティグループとは, インスタンスへの情報の流れを制御するファイアウォールである.

たメインのライブラリのうち, Web アプリケーションフレームワークとして Sinatra を用い, そして, センサーデータを格納するデータベースに SQlite3 を導入した.

4.4.4 Web ページの作成

完成したデザインは以下の画像[*15]である. なお, Ruby のスクリプトを埋め込むため, 拡張子は.html から.erb としている. なお, Ruby のスクリプトを読み込み続ける必要があるので, HTML ファイルの head タグの中に, 一分毎に再読み込みを行う meta 要素を追加している. そして, 温度と湿度の数値の div にはインスタンス変数を埋め込んでいる. センサーデータの最新の値をインスタンス変数に当てることで, 一分毎に最新データを表示することが出来る. また, 背景画像は, 最新のセンサーデータを条件式にはめて適切な画像を表示するようにしている. 時間に関しては, JavaScript を用いて現在の時刻を取得している.

図 6: Web ページの画像

4.4.5 サーバサイドスクリプト

読み込みのリクエストが来たときは, データベースに格納されているデータのうち最新のデータをインスタンス変数に入れる. そのうち, 明るさデータの方は, 一定の区切りを設けて晴れ・曇り・夜の画像を表示する. 画像は夜以外は複数枚用意している. GPIO データの方は, 常に雨雫の画像を表示している状態で, 値が一定超えると透明度の値を増減させることで表示状態を変える. 温度と湿度データは, インスタンス変数に入れるだけで表示される.

[*15]https://www.pakutaso.com/

データと画像を用意出来たら, erb ファイルが読み込まれる. Raspberry Pi からセンサーデータが POST されたときの処理はシンプルで, JSON 形式で送られてきたデータをハッシュの形で取得して, データベースに格納していくだけである.

4.4.6 実装面の展望

今後の展望として, リロードの処理をリアルタイム更新にすることで, より正確な情報を提供できると考える. また, 天気の判定方法にセンサを追加をすることや, 実際に天気予報のデータを API で取得して, 条件式にはめて表示するのも一つだと考える. 天気予報で得られるデータの対象地域は広いことため, 現地では雨が降っていないといったこともあり得る. このことから, センサで現地のデータを取ることで実際の窓として実現できるということである. しかし, この逆の考え方もできる. センサの誤作動値を取得して雨と認識してしまったが実際は降っていない, といったことである. この場合, 広域で晴れと出ていれば, センサの値を切り捨てて晴れと表示することで対策できる. より正確な天気を届けられるために, これらを取り入れていくのも良いのではないかと考える.

5　おわりに

5.1　実装に関する考察

文責：井上　諒也

本来のシステムではAWSのKinesis Data StreamやKinesis Firehoseを用いる予定だった．今回では実装を優先したため, Raspberry PiからAWS EC2を用いて構築したWebサーバに直接データを送る仕様に変更した．Kinesis Data Streamを用いることで大規模なデータのリアルタイム処理に対して, データの移動に負荷を最小限に抑えることができる．Delivery SystemであるKinesis Firehoseを用いることで, パケットに送信されるデータが暗号化されるなどセキュリティの問題においてもある程度, 対策がなされる．今回は小規模なデータの活用であったため, 用途に合わせたシステムの設計が必要であると再確認した．プロジェクトとしての目標はシステムの構築であったため, この目標に対しては, おおむね達成できたといえる．

問題点として挙げられるのは, Bluetooth接続しているMESHの数が多く, Raspberry PiとMESHの接続状況が不安定になる．解決策として, MESHの数を減らすか, 接続が切断されると自動的に”hcitool lescan”コマンドを実行し, 再接続を試みる案が挙げられる．また, 実用化を行うとなった場合に電源問題と降水量を推定するためのデータを精密に取得する必要がある為, センサーの設置方法については再考が必要である．

5.2　IoTの展望

文責：水野佑哉

この項ではIoT技術自体の将来の展望について記す．IoTは様々な情報を収集，発信して見て使えるようにすることでより生活を便利にできる技術である．現在盛んに開発が行われている自動車の自動運転技術にもとても必要な技術であり，その他にも今あるものをより安全で簡単に使用できるようにすることのできる技術であると考えられる．そのため，今後も今以上に企業などが力を入れて開発する分野であると言える．IoTはうまく利用できればとても便利な技術であるが，物がインターネットにつながるということはクラッキングの危険性が隣り合わせになるということでもある．これからのIoT開発はもちろんより活発になるが，セキュリティ面など慎重に進められていく分野でもあると言えるだろう．

5.3　プロジェクトの感想

文責：井上　諒也

本プロジェクトではIoTの定義に基づいたシステムの構築を「部室に窓を設置する」というテーマを用いて行った. 6月度にOB・OGの方々から寄贈されたMESHを活用し, 班員の知識と技術力の向上に繋げることができた. またMESHだけではなく電子工作やCloudService, サーバ構築, Webページ制作など様々な分野と関連させることができたため, それらが班員の好奇心を掻き立てたことに間違いはないだろう. 実装過程において班員以外の方が助言をしてくださったりなど, ご尽力いただいたので, ここで感謝の意を申し上げたい.

参考文献

[1] mobile backend「IoTの最新動向＆コラム世界が変わる！モノのインターネット」(2016年04月21日)〔http://iot.mb.cloud.nifty.com/iotcolumn〕(最終閲覧日：2018年2月4日)

[2] 「平成 27 年版情報通信白書」〔http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/pdf/n5400000.pdf〕平成二十七年度

[3] トーマス･H･ダベンポート(Thomas H. Davenport)「モノのインターネット(IoT)」〔https://www.sas.com/ja_jp/insights/big-data/internet-of-things.html〕(最終閲覧日：2018年2月4日)

[4] MONOWIRELESS「IoTとは? | IoT：Internet of Things(モノのインターネット)の意味」〔https://mono-wireless.com/jp/tech/Internet_of_Things.html〕(最終閲覧日：2018年2月4日)

[5] 「MESH公式サイト 「つくれる、学べる、楽しめる アイデアを形にできる IoT ブロック」」〔http://meshprj.com/jp/〕(最終閲覧日：2018年2月3日)

[6] 「AWS JP 公式」〔https://aws.amazon.com/jp/〕(最終閲覧日：2018年2月3日)

[7] 「Sony MESHのButtonタグをハック」〔http://hikolab.com/blog/?p=99〕(最終閲覧日：2017年12月1日)

[8] 「Ssandeepmistry/noble」〔https://github.com/sandeepmistry/noble〕(最終閲覧日：2017年12月1日)